Robust Satisfaction of Temporal Logic Specifications via Reinforcement Learning

Austin Jones[1], Derya Aksaray[2], Zhaodan Kong[3], Mac Schwager[4], and Calin Belta[2,5]

Abstract— We consider the problem of steering a system with unknown, stochastic dynamics to satisfy a rich, temporally-layered task given as a signal temporal logic formula. We represent the system as a Markov decision process in which the states are built from a partition of the statespace and the transition probabilities are unknown. We present provably convergent reinforcement learning algorithms to maximize the probability of satisfying a given formula and to maximize the average expected robustness, i.e., a measure of how strongly the formula is satisfied. We demonstrate via a pair of robot navigation simulation case studies that reinforcement learning with robustness maximization performs better than probability maximization in terms of both probability of satisfaction and expected robustness.

I. INTRODUCTION

We consider the problem of controlling a system with unknown, stochastic dynamics, i.e., a "black box", to achieve a complex, time-sensitive task. An example is controlling a noisy aerial vehicle with partially known dynamics to visit a pre-specified set of regions in some desired order while avoiding hazardous areas. We consider tasks given as temporal logic (TL) formulae [2], an extension of first order Boolean logic that can be used to reason about how the state of a system evolves over time. When a stochastic dynamical model is known, there exist algorithms to find control policies for maximizing the probability of achieving a given TL specification [18], [17], [23], [13] by planning over stochastic abstractions [12], [1], [17]. However, only a handful of papers have considered the problem of enforcing TL specifications to a system with unknown dynamics. Passive [3] and active [21], [9] reinforcement learning has been used to find a policy that maximizes the probability of satisfying a given linear temporal logic formula.

In this paper, in contrast to the above works on reinforcement learning which use propositional temporal logic, we use signal temporal logic (STL), a rich predicate logic that can be used to describe tasks involving bounds on physical parameters and time intervals [7]. An example of such a property is "Within t_1 seconds, a region in which y is less than π_1 is reached, and regions in which y is larger than π_2 are avoided for t_2 seconds." STL admits a continuous measure called *robustness degree* that quantifies how strongly a given sample path exhibits an STL property as a real number rather than just providing a *yes* or *no* answer [8], [7]. This measure enables the use of continuous optimization methods to solve inference (e.g., [10], [11], [14]) or formal synthesis problems (e.g., [20]) involving STL.

One of the difficulties in solving problems with TL formulae is the history-dependence of their satisfaction. For instance, if the specification requires visiting region A before region B, whether or not the system should steer towards region B depends on whether or not it has previously visited region A. For linear temporal logic (LTL) formulae with time-abstract semantics, this history-dependence can be broken by translating the formula to a deterministic Rabin automaton (DRA), a model that automatically takes care of the history-dependent "book-keeping" [4], [21]. In the case of STL, such a construction is difficult due to the time-bounded semantics. We circumvent this problem by defining a fragment of STL such that the progress towards satisfaction is checked with some finite number τ of state measurements. We thus define an MDP, called the τ-MDP whose states correspond to the τ-step history of the system. The inputs to the τ-MDP are a finite collection of control actions.

We use a reinforcement learning strategy called Q-learning [24], in which a policy is constructed by taking actions, observing outcomes, and reinforcing actions that improve a given reward. Our algorithms either maximize the probability of satisfying a given STL formula, or maximize the expected robustness with respect to the given STL formula. These procedures provably converge to the optimal policy for each case. Furthermore, we propose that maximizing expected robustness is typically more effective than maximizing probability of satisfaction. We prove that in certain cases, the policy that maximizes expected robustness also maximizes the probability of satisfaction. However, if the given specification is not satisfiable, the probability maximization will return an arbitrary policy, while the robustness maximization will return a policy that gets as close to satisfying the policy as possible. Finally, we demonstrate through simulation case studies that the policy that maximizes expected robustness in some cases gives better performance in terms of both probability of satisfaction and expected robustness when fewer training episodes are available.

[1]Author is with Mechanical Engineering and Electrical Engineering, Georgia Institute of Technology, Atlanta, GA, USA. austinjones@gatech.edu

[2]Authors are with Mechanical Engineering, Boston University, Boston, MA, USA. {cbelta,daksaray}@bu.edu

[3]Author is with Mechanical and Aerospace Engineering, University of California Davis, Davis, CA, USA. zdkong@ucdavis.edu

[4] Author is with Aeronautics and Astronautics, Stanford University, Stanford, CA, USA. schwager@stanford.edu

[5]Author is with Systems Engineering, Boston University, Boston, MA, USA.

II. SIGNAL TEMPORAL LOGIC(STL)

STL is defined with respect to continuously valued signals. Let $\mathscr{F}(A,B)$ denote the set of mappings from A to B and define a *signal* as a member of $\mathscr{F}(\mathbb{N},\mathbb{R}^n)$. For a signal s, we denote s^t as the value of s at time t and $s^{t_1:t_2}$ as the sequence of values $s^{t_1}s^{t_1+1}\dots s^{t_2}$. Moreover, we denote $s[t]$ as the *suffix* from time t, i.e., $s[t] = \{s^{t'}|t' \ge t\}$.

In this paper, the desired mission specification is described by an STL fragment with the following *syntax* :

$$\begin{aligned} \phi &:= F_{[0,T]}\psi|G_{[0,T]}\psi, \\ \psi &:= f(s) \le d|\neg\varphi|\varphi_1 \wedge \varphi_2|\varphi_1 U_{[a,b)}\varphi_2, \end{aligned} \quad (1)$$

where T is a finite time bound, ϕ,ψ, and φ are STL formulae, a and b are non-negative real-valued constants, and $f(s) < d$ is a predicate where s is a signal, $f \in \mathscr{F}(\mathbb{R}^n,\mathbb{R})$ is a function, and $d \in \mathbb{R}$ is a constant. The Boolean operators $\neg$ and $\wedge$ are negation ("not") and conjunction ("and"), respectively. The other Boolean operators are defined as usual. The temporal operators F, G, and U stand for "Finally (eventually)" , "Globally (always)", and "Until", respectively. Note that in this paper, we use a discrete-time version of STL rather than the typical continuous-time formulation.

The *semantics* of STL is recursively defined as

$$\begin{aligned} s[t] &\models (f(s) < d) &\text{iff}&\quad f(s^t) < d \\ s[t] &\models \phi_1 \wedge \phi_2 &\text{iff}&\quad s[t] \models \phi_1 \text{ and } s[t] \models \phi_2 \\ s[t] &\models \phi_1 \vee \phi_2 &\text{iff}&\quad s[t] \models \phi_1 \text{ or } s[t] \models \phi_2 \\ s[t] &\models G_{[a,b]}\phi &\text{iff}&\quad s[t'] \models \phi \\ & & & \forall t' \in [t+a,t+b) \\ s[t] &\models F_{[a,b]}\phi &\text{iff}&\quad \exists t' \in [t+a,t+b) \\ & & & \text{s.t. } s[t'] \models \phi \\ s[t] &\models \phi_1 U_{[a,b]}\phi_2 &\text{iff}&\quad \exists t' \in [t+a,t+b) \\ & & & \text{s.t. } s[t''] \models \phi_1 \forall t'' \in [t,t') \\ & & & \text{and } s[t'] \models \phi_2. \end{aligned}$$

In plain English, $F_{[a,b]}\phi$ means "within a and b time units in the future, ϕ is true," $G_{[a,b]}\phi$ means "for all times between a and b time units in the future ϕ is true," and $\phi_1 U_{[a,b]}\phi_2$ means "There exists a time c between a and b time units in the future such that ϕ_1 is true until c and ϕ_2 is true at c." STL is equipped with a *robustness degree* [8], [7] (also called "degree of satisfaction") that quantifies how well a given signal s satisfies a given formula ϕ. The robustness is calculated recursively according to the *quantitative semantics*

$$\begin{aligned} r(s,(f(s) < d),t) &= d - f(s^t) \\ r(s,\phi_1 \wedge \phi_2,t) &= \min\big(r(s,\phi_1,t),r(s,\phi_2,t)\big) \\ r(s,\phi_1 \vee \phi_2,t) &= \max\big(r(s,\phi_1,t),r(s,\phi_2,t)\big) \\ r(s,G_{[a,b]}\phi,t) &= \min_{t'\in[t+a,t+b]} r(s,\phi,t') \\ r(s,F_{[a,b]}\phi,t) &= \max_{t'\in[t+a,t+b]} r(s,\phi,t'), \\ r(s,\phi_1 U_{[a,b]}\phi_2,t) &= \sup_{t'\in[t+a,t+b]}\Big(\min\big(r(\phi_2,s,t'), \\ & \quad \inf_{t''\in[t,t']} r(\phi_1,s,t'')\big)\Big). \end{aligned}$$

We use $r(s,\phi)$ to denote $r(s,\phi,0)$. If $r(s,\phi)$ is large and positive, then s would have to change by a large deviation in order to violate ϕ. Similarly, if $r(s,\phi)$ is large in absolute value and negative, then s strongly violates ϕ.

Similar to [6], let $hrz(\phi)$ denote the *horizon length* of an STL formula ϕ. The horizon length is the required number of samples to resolve any (future or past) requirements of ϕ. The horizon length can be computed recursively as

$$\begin{aligned} hrz(p) &= 0, \\ hrz(\neg\phi) &= hrz(\phi), \\ hrz(\phi_1 \vee \phi_2) &= \max\{hrz(\phi_1),hrz(\phi_2)\}, \\ hrz(\phi_1 \wedge \phi_2) &= \max\{hrz(\phi_1),hrz(\phi_2)\}, \\ hrz(\phi_1 U_{[a,b]}\phi_2) &= \max\{hrz(\phi_1)+b-1,hrz(\phi_2)+b\}, \end{aligned} \quad (2)$$

where ϕ,ϕ_1,ϕ_2 are STL formulae.

Example 1: Consider the robot navigation problem illustrated in Figure 1(a). The specification is "Visit Regions A or B and visit Regions C or D every 4 time units along a mission horizon of 100 units." Let $s(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}^T$, where x and y are the $x-$ and $y-$ components of the signal s. This task can be formulated in STL as

$$\begin{aligned} \phi = &\ G_{[0,100)} \quad \psi \\ \psi = &\ \Big(F_{[0,4)} \quad \big((x > 2 \wedge x < 3 \wedge y > 2 \wedge y < 3) \\ & \qquad\quad \vee (x > 4 \wedge x < 5 \wedge y > 4 \wedge y < 5)\big) \quad (3) \\ & \wedge F_{[0,4)} \quad \big((x > 2 \wedge x < 3 \wedge y > 4 \wedge y < 5) \\ & \qquad\quad \vee (x > 4 \wedge x < 5 \wedge y > 2 \wedge y < 3)\big)\Big). \end{aligned}$$

Figure 1(a) shows two trajectories of the system beginning at the initial location of R and ending in region C that each satisfies the inner specification ψ given in (3). Note that s_2 barely satisfies ψ, as it only slightly penetrates region A, while s_1 appears to satisfy it strongly, as it passes through the center of region A and the center of region C. The robustness degrees confirm this: $r(s_1,\psi) = 0.3$ while $r(s_2,\psi) = 0.05$.

The horizon length of the inner specification ψ of (3) is

$$hrz(\psi) = \max\big(4 + \max(0,0), 4 + \max(0,0)\big) = 4.$$

∎

III. MODELS FOR REINFORCEMENT LEARNING

For a system with unknown and stochastic dynamics, a critical problem is how to synthesize control to achieve a desired behavior. A typical approach is to discretize the state and action spaces of the system and then use a *reinforcement learning strategy*, i.e., by learning how to take actions through trial and error interactions with an unknown environment [22]. In this section, we present models of systems that are amenable for reinforcement learning to enforce temporal logic specifications. We start with a discussion on the widely used LTL before introducing the particular model that we will use for reinforcement learning with STL.

A. Reinforcement Learning with LTL

One approach to the problem of enforcing LTL satisfaction in a stochastic system is to partition the statespace and design control primitives that can (nominally) drive the system from one region to another. These controllers, the stochastic dynamical model of the system, and the quotient obtained from the partition are used to construct a Markob decision process (MDP), called a bounded parameter MDP or BMDP,

whose transition probabilities are interval-valued [1]. These BMDPs can then be composed with a DRA constructed from a given LTL formula to form a product BMDP. Dynamic programming (DP) can then be applied over this product MDP to generate a policy that maximizes the probability of satisfaction. Other approaches to this problem include aggregating the states of a given quotient until an MDP can be constructed such that the transition probability can be considered constant (with bounded error) [16]. The optimal policy can be computed over the resulting MDP using DP [15] or approximate DP, e.g., actor-critic methods [5].

Thus, even when the stochastic dynamics of a system are known and the logic that encodes constraints has time-abstract semantics, the problem of constructing an abstraction of the system that is amenable to control policy synthesis is difficult and computationally intensive. Reinforcement learning methods for enforcing LTL constraints make the assumption that the underlying model under control is an MDP [3], [21], [9]. Implicitly, these procedures compute a frequentist approximation of the transition probabilities that asymptotically approaches the true (unknown) value as the number of observed sample paths increases. Since this algorithm doesn't explicitly rely on any *a priori* knowledge of the transition probability, it could be applied to an abstraction of a continuous-space system that is built from a proposition-preserving partition. In this case, the uncertainty on the motion described by intervals in the BMDP that is reduced via computation would instead be described by complete ignorance that is reduced via learning. The resulting policy would map regions of the statespace to discrete actions that will optimally drive the real-valued state of the system to satisfy the given LTL specification. Different partitions will result in different policies. In the next section, we extend the above observation to derive a discrete model that is amenable for reinforcement learning for STL formulae.

B. Reinforcement learning with STL: τ-MDP

In order to reduce the search space of the problem, we partition the statespace of the system to form the quotient graph $\mathscr{G} = (\Sigma, E)$, where Σ is a set of discrete states corresponding to the regions of the statespace and E corresponds to the set of edges. An edge between two states σ and σ' exists in E if and only if σ and σ' are neighbors (share a boundary) in the partition. In our case, since STL has time-bounded semantics, we cannot use an automaton with a time-abstract acceptance condition (e.g., a DRA) to check its satisfaction. In general, whether or not a given trajectory $s^{0:T}$ satisfies an STL formula would be determined by directly using the qualitative semantics. The STL fragment (1) consists of a sub-formula ψ with horizon length $hrz(\psi) = \tau$ that is modified by either a $F_{[0,T)}$ or $G_{[0,T)}$ temporal operator. This means that in order to update at time t whether or not the given formula ϕ has been satisfied or violated, we can use the τ previous state values $s^{t-\tau+1:t}$ For this reason, we choose to learn policies over an MDP with finite memory, called a τ-MDP, whose states correspond to sequences of length τ of regions in the defined partition.

Example 1 (cont'd): Let the robot evolve according to the discrete-time Dubins dynamics

$$\begin{aligned} x^{t+1} &= x^t + v\delta^t \cos\theta^t \\ y^{t+1} &= y^t + v\delta^t \sin\theta^t, \end{aligned} \quad (4)$$

where x^t and y^t are the x and y coordinates of the robot at time t, v is its forward speed, δ^t is a time interval, and the robot's orientation is given by θ^t. The control primitives in this case are given by $Act = \{up, down, left, right\}$ which correspond to the directions on the grid. Each (noisy) control primitive induces a distribution with support $\theta_{des} \pm \Delta\theta$, where θ_{des} is the orientation where the robot is facing the desired cell. When a motion primitive is enacted, the robot rotates to an angle θ^t drawn from the distribution and moves along that direction for δ^t time units. The partition of the statespace and the induced quotient $\mathscr{G}$ are shown in Figures 1(b) and 1(c), respectively. A state $\sigma_{(i,j)}$ in the quotient (Figure 1(c)) represents the region in the partition of the statespace (Figure 1(b)) with the point (i,j) in the lower left hand corner. ∎

Definition 1: Given a quotient of a system $\mathscr{G} = (\Sigma, E)$ and a finite set of actions Act, a τ-*Markov Decision Process (τ-MDP)* is a tuple $\mathscr{M}_\tau = \langle \mathscr{S}, Act, \mathscr{P} \rangle$, where

- $\mathscr{S} \subseteq (\Sigma \cup \varepsilon)^\tau$ is the set of finite states, where ε is the empty string. Each state $\sigma_\tau \in \mathscr{S}$ corresponds to a τ-horizon (or shorter) path in $\mathscr{G}$. Shorter paths of length $n < \tau$ (representing the case in which the system has not yet evolved for τ time steps) have ε prepended $\tau - n$ times.
- $\mathscr{P} : \mathscr{S} \times Act \times \mathscr{S} \to [0,1]$ is a probabilistic transition relation. $\mathscr{P}(\sigma_\tau, a, \sigma'_\tau)$ can be positive only if the first $\tau - 1$ states of σ'_τ are equal to the last $\tau - 1$ states of σ_τ and there exists an edge in $\mathscr{G}$ between the final state of σ_τ and the final state of σ'_τ.

We denote the state of the τ-MDP at time t as σ_τ^t.

Definition 2: Given a trajectory $s^{t-\tau+1:t}$ of the original system, we define its induced *trace* in the τ-MDP $\mathscr{M}_\tau$ as $Tr(s^{t-\tau+1:t}) = \sigma^{t-\tau+1:t} = \sigma_\tau^t$. That is, σ_τ^t corresponds to the previous τ regions of the statespace that the state has resided in from time $t - \tau + 1$ to time t.

The construction of a τ-MDP from a given quotient and set of actions is straightforward. The details are omitted due to length constraints. We make the following key assumptions on the quotient and the resulting τ-MDP:

- The defined control actions Act will drive the system either to a point in the current region or to a point in a neighboring region of the partition, e.g., no regions are "skipped".
- The transition relation $\mathscr{P}$ is Markovian.

For every τ state σ_τ^t, there exists a continuous set of sample paths $\{s^{t-\tau+1:t}\}$ whose traces could be that state. The dynamics of the underlying system produces an unknown distribution $p(s^{t-\tau+1:t}|Tr(s^{t-\tau+1:t}) = \sigma_\tau^t)$. Since the robustness degree is a function of sample paths of length τ and an STL formula ψ, we can define a distribution $p(r(s^{t-\tau+1:t}, \psi)|Tr(s^{t-\tau+1:t}) = \sigma_\tau^t)$.
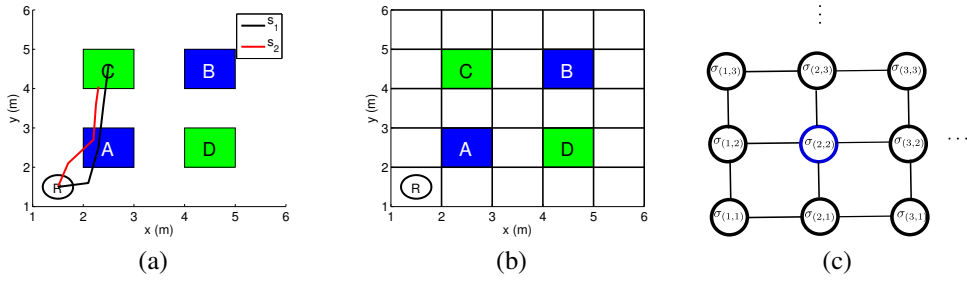
Fig. 1: (a) Example of robot navigation problem. (b) Partitioned space. (c) Subsection of the quotient.

Example 1 (cont'd): Figure 2 shows a portion of the τ-MDP constructed from Figure 1. The states in $\mathscr{M}_4$ are labeled with the corresponding sample paths of length 4 in $\mathscr{G}$. The green and blue σ's in the states in $\mathscr{M}_4$ correspond to green and blue regions from Figure 1.

IV. PROBLEM FORMULATION

In this paper, we address the following two problems.

Problem 1 (Maximizing Probability of Satisfaction): Let $\mathscr{M}_\tau$ be a τ-MDP as described in the previous section. Given an STL formula ϕ with syntax (1), find a policy $\mu_{mp}^* \in \mathscr{F}(\mathscr{S} \times \mathbb{N}, Act)$ such that

$$\mu_{mp}^* = \underset{\mu \in \mathscr{F}(\mathscr{S} \times \mathbb{N}, Act)}{\arg\max} Pr_{s^{0:T}}[s^{0:T} \models \phi] \quad (5)$$

Problem 2 (Maximizing Average Robustness): Let $\mathscr{M}_\tau$ be as defined in Problem 1. Given an STL formula ϕ with syntax (1), find a policy $\mu_{mr}^* \in \mathscr{F}(\mathscr{S}, Act)$ such that

$$\mu_{mr}^* = \underset{\mu \in \mathscr{F}(\mathscr{S} \times \mathbb{N}, Act)}{\arg\max} E_{s^{0:T}}[r(s^{0:T}, \phi)] \quad (6)$$
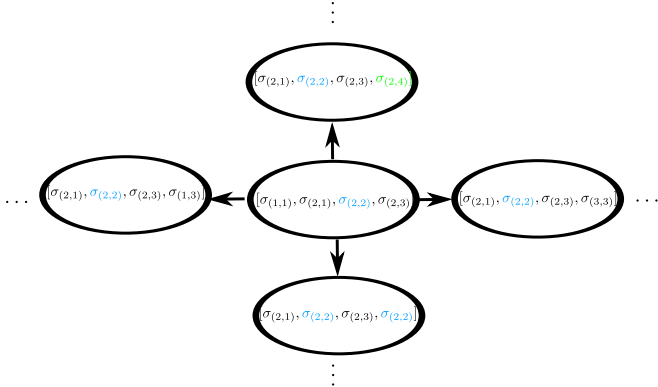

Fig. 2: Part of the τ-MDP constructed from the robot navigation MDP shown in Figure 1

Problems 1 and 2 are two alternate solutions to enforce a given STL specification. The policy found by Problem 1, i.e. μ_{mp}^*, maximizes the chance that ϕ will be satisfied, while the policy found by Problem 2, i.e. μ_{mr}^*, drives the system to satisfy ϕ as strongly as possible on average. Problems similar to (5) have already been considered in the literature (e.g., [9], [21]). However, Problem 2 is a novel formulation that provides some advantages over Problem 1. As we show in Section V, for some special systems, μ_{mr}^* achieves the

same probability of satisfaction as μ_{mp}^*. Furthermore, if ϕ is not satisfiable, any arbitrary policy could be a solution to Problem 1, as all policies will result in a satisfaction probability of 0. If ϕ is unsatisfiable, Problem 2 yields a solution that attempts to get as close as possible to satisfying the formula, as the optimal solution will have an average robustness value that is least negative.

The forms of the objective functions differ for the two different types of formula, $\phi = F_{[0,T)}\psi$ and $\phi = G_{[0,T)}\psi$. *Case 1:* Consider an STL formula $\phi = F_{[0,T)}\psi$. In this case, the objective function in (5) can be rewritten as

$$Pr_{s^{0:T}}[\exists t = \tau, \dots, T - \tau \text{ s.t. } s^{t-\tau+1:t} \models \psi], \quad (7)$$

and the objective function in (6) can be rewritten as

$$E_{s^{0:T}}\left[\max_{t=\tau,\dots,T-\tau} r(s^{t-\tau+1:t}, \psi)\right]. \quad (8)$$

Case 2: Now, consider an STL formula $\phi = G_{[0,T)}\psi$. The objective function in (5) can be rewritten as

$$Pr_{s^{0:T}}[\forall t = \tau, \dots, T - \tau, s^{t-\tau:t} \models \psi], \quad (9)$$

and the objective function in (6) can be rewritten as

$$E_{s^{0:T}}\left[\min_{t=\tau,\dots,T-\tau} r(s^{t-\tau+1:t}, \psi)\right]. \quad (10)$$

V. MAXIMIZING EXPECTED ROBUSTNESS VS. MAXIMIZING PROBABILITY OF SATISFACTION

Here, we demonstrate that the solution to (6) subsumes the solution to (5) for a certain class of systems. Due to space limitations, we only consider formulae of the type $\phi = F_{[0,t)}\psi$. Let $\mathscr{M}_\tau = (\mathscr{S}_\tau, \mathscr{P}_\tau, Act)$ be a τ-MDP. For simplicity, we make the following assumption on $\mathscr{S}_\tau$.

Assumption 1: For every state $\sigma_\tau \in \mathscr{S}_\tau$, either every trajectory $s^{t+\tau-1:t}$ whose trace is σ_τ satisfies ψ, denoted $\sigma_\tau \models \psi$, or every trajectory that passes through the sequence of regions associated with σ_τ does not satisfy ψ, denoted $\sigma_\tau \not\models \psi$.

Assumption 1 can be enforced in practice during partitioning. We define the set

$$A = \{\sigma_\tau \in \mathscr{S}_\tau | \sigma_\tau \models \psi\}. \quad (11)$$

Definition 3: The signed graph distance of a τ-state $\sigma_\tau^i \in \mathscr{S}$ to a set $X \subseteq \mathscr{S}$ is

$$d(\sigma_\tau^i, X) = \begin{cases} \min_{\sigma_\tau^j \in X} l(\sigma_\tau^i, \sigma_\tau^j) & \sigma_\tau^i \notin X \\ -\min_{\sigma_\tau^j \in \mathscr{S}_\tau \setminus X} l(\sigma_\tau^j, \sigma_\tau^i) & \sigma_\tau^i \in X \end{cases} \quad (12)$$

where $l(\sigma_\tau^i, \sigma_\tau^j)$ is the length of the shortest path from σ_τ^i to σ_τ^j.

We also make the following two assumptions.

Assumption 2: For any signal $s^{t-\tau+1:t}$ such that $Tr(s^{t-\tau+1:t}) \in \mathscr{S}_\tau$, let $r(s^{t-\tau+1:t}, \psi)$ be bounded from below by R_{min} and from above by R_{max}.

Assumption 3: Let $D_{\sigma_\tau}(\delta) = Pr[r(s^{t:t+\tau}, \psi) > \delta | Tr(s^{t:t+\tau}) = \sigma_\tau]$. For any two states,

$$d(\sigma_\tau^i, A) < d(\sigma_\tau^j, A) \Rightarrow D_{\sigma_\tau^i}(\delta) \geq D_{\sigma_\tau^j}(\delta) \; \forall \delta \in [R_{min}, R_{max}] \tag{13}$$

Now we define the policies μ_{mp}^* and μ_{mr}^* over $\mathscr{M}_\tau$ as

$$\mu_{mp}^* = \underset{\mu \in \mathscr{F}(\mathscr{S} \times \mathbb{N}, Act)}{\arg\max} Pr_{\sigma_\tau^{0:T}}\left[\exists t \in [0, T] \text{ s.t. } \sigma_\tau^t \models \psi\right] \tag{14}$$

$$\mu_{mr}^* = \underset{\mu \in \mathscr{F}(\mathscr{S} \times \mathbb{N}, Act)}{\arg\max} E_{\sigma_\tau^{0:T}}\left[\max_{t=0,...,T} r(\sigma_\tau^t, \psi)\right] \tag{15}$$

Proposition 1: If Assumptions 1,2, and 3 hold, then the policy μ_{mr}^* maximizes the expected probability of satisfaction.

Proof: Given any policy μ, its associated reachability probability can be defined as

$$Pr_\mu(\sigma_\tau) = Pr_\mu\left[\sigma_\tau = \underset{\sigma_\tau^0, ..., \sigma_\tau^{T-\tau}}{\arg\min} d(\sigma_\tau, A)\right]. \tag{16}$$

Let $I(.)$ be the indicator function such that $I(B)$ is 1 if B is true and 0 if B is false. By definition, the expected probability of satisfaction for a given policy μ is

$$\begin{aligned} EPS(\mu) &= E\left[I(\exists 0 < k < T - \tau \text{ s.t. } \sigma_\tau^k \models \psi)\right] \\ &= \sum_{\sigma_\tau \in \mathscr{S}_\tau} Pr_\mu(\sigma_\tau) I(\sigma_\tau \in A) \\ &= \sum_{\sigma_\tau \in A} Pr_\mu(\sigma_\tau). \end{aligned} \tag{17}$$

Also, the expected robustness of policy μ becomes

$$\begin{aligned} ER(\mu) &= E\left[\max_{k=0,...,T-\tau} r(\sigma_\tau^k, \psi)\right] \\ &= \int_0^{R_{max}} Pr\left[\max_{k=0,...,T-\tau} r(\sigma_\tau^k, \psi) > x\right] dx + \\ &\quad \int_{R_{min}}^0 1 - Pr\left[\max_{k=0,...T-\tau} r(\sigma_\tau^k, \psi) > x\right] dx \\ &= \int_0^{R_{max}} Pr\left[\max_{k=0,...,T-\tau} r(\sigma_\tau^k, \psi) > x\right] dx - \\ &\quad \int_{R_{min}}^0 Pr\left[\max_{k=0,...,T-\tau} r(\sigma_\tau^k, \psi) > x\right] dx - R_{min} \\ &= \int_0^{R_{max}} \sum_{\sigma_\tau \in \mathscr{S}_\tau} Pr_\mu(\sigma_\tau) D_{\sigma_\tau}(x) dx - \\ &\quad \int_{R_{min}}^0 \sum_{\sigma_\tau \in \mathscr{S}_\tau} Pr_\mu(\sigma_\tau) D_{\sigma_\tau}(x) dx - R_{min} \\ &= \sum_{\sigma_\tau \in A} Pr_\mu(\sigma_\tau) \int_0^{R_{max}} D_{\sigma_\tau}(x) dx - \\ &\quad \sum_{\sigma_\tau \notin A} Pr_\mu(\sigma_\tau) \int_{R_{min}}^0 D_{\sigma_\tau}(x) dx - R_{min}. \end{aligned} \tag{18}$$

Since R_{min} is constant, maximizing (18) is equivalent to

$$\max_\mu \left(\sum_{\sigma_\tau \in A} Pr_\mu(\sigma_\tau) \int_0^{R_{max}} D_{\sigma_\tau}(x) dx - \sum_{\sigma_\tau \notin A} Pr_\mu(\sigma_\tau) \int_{R_{min}}^0 D_{\sigma_\tau}(x) dx \right) \tag{19}$$

Let p be the satisfaction probability such that $p = \sum_{\sigma_\tau \in A} Pr_\mu(\sigma_\tau)$. Then, we can rewrite the objective in (19) as

$$\begin{aligned} J(\mu) &= p \sum_{\sigma_\tau \in A} Pr_\mu\left[\sigma_\tau = \underset{\sigma_\tau^0, ..., \sigma_\tau^{T-\tau}}{\arg\min} d(\sigma_\tau, A) | \sigma_\tau \in A\right] \\ &\quad \times \int_0^{R_{max}} D_{\sigma_\tau}(x) dx \\ &\quad - (1-p) Pr_\mu\left[\sigma_\tau = \underset{\sigma_\tau^0, ..., \sigma_\tau^{T-\tau}}{\arg\min} d(\sigma_\tau, A) | \sigma_\tau \notin A\right] \\ &\quad \times \int_{R_{min}}^0 D_{\sigma_\tau}(x) dx. \end{aligned} \tag{20}$$

Now,

$$\begin{aligned} \frac{\partial J(\mu)}{\partial p} &= \sum_{\sigma_\tau \in A} Pr_\mu\left[\sigma_\tau = \underset{\sigma_\tau^0, ..., \sigma_\tau^{T-\tau}}{\arg\min} d(\sigma_\tau, A) | \sigma_\tau \in A\right] \\ &\quad \times \int_0^{R_{max}} D_{\sigma_\tau}(x) dx \\ &\quad + Pr_\mu\left[\sigma_\tau = \underset{\sigma_\tau^0, ..., \sigma_\tau^{T-\tau}}{\arg\min} d(\sigma_\tau, A) | \sigma_\tau \notin A\right] \\ &\quad \times \int_{R_{min}}^0 D_{\sigma_\tau}(x) dx \\ &> 0 \end{aligned} \tag{21}$$

Thus, any policy μ increasing $J(\mu)$ also leads to an increase in p. Since increasing $J(\mu)$ is equivalent to increasing $ER(\mu)$, then we can conclude that the policy that maximizes the robustness also achieves the maximum satisfaction probability. $\blacksquare$

VI. Control Synthesis to Maximize Robustness

A. Policy Generation through Q-Learning

Since we do not know the dynamics of the system under control, we cannot *a priori* predict how a given control action will affect the evolution of the system and hence its progress towards satisfying/dissatisfying a given specification. Thus, we use the well-known paradigm of *reinforcement learning* to learn policies to solve Problems 1 and 2. In reinforcement learning, the system takes actions and records the rewards associated with the state-action pair. These rewards are then used to update a feedback policy that maximizes the expected gathered reward. In our cases, the rewards that we collect over $\mathscr{M}_\tau$ are related to whether or not ψ is satisfied (Problem 1) or how robustly ψ is satisfied/violated (Problem 2).

Our solutions to these problems rely on a *Q-learning* formulation [24]. Let $R(\sigma_\tau^t, a)$ be the reward collected when action $a \in Act$ was taken in state $\sigma_\tau^t \in \mathscr{S}$. Define the function $Q : \mathscr{S} \times Act \times \mathbb{N}$ as

$$\begin{aligned} Q(\sigma_\tau^{T-t}, a, t) &= R(\sigma_\tau^{T-t}, a) + \\ &\quad \max_{\{\mu_l \in\}_{l=T-t-1}^T} E\left[\sum_{l=T-t-1}^T R(\sigma_\tau^l, \mu_l(\sigma_\tau^l))\right] \\ &= R(\sigma_\tau^{T-t}, a) + \max_{a' \in Act} Q(\sigma_\tau^{T-t+1}, a', t-1). \end{aligned} \tag{22}$$

For an optimization problem with a cumulative objective function of the form

$$\sum_{l=\tau:T} R(\sigma_\tau^l, a^l), \tag{23}$$

the optimal policy $\mu^* \in \mathscr{F}(\mathscr{S}, Act)$ can be found by

$$\mu^*(\sigma_\tau^t, T-t) = \underset{a \in Act}{\arg\max} Q(\sigma_\tau^t, a, T-t). \tag{24}$$

Applying the update rule

$$Q_{t+1}(\sigma_\tau^t, a^t, T-t) = \quad (1-\alpha_t)Q_t(\sigma_\tau^t, a^t, T-t) + \\ \alpha_t[R(\sigma_\tau^t, a^t) + \gamma \max_{a' \in A} Q_t(\sigma_\tau^{t+1}, a')]$$

(25)

where $0 < \gamma < 1$ will cause Q_t converges to Q w.p. 1 as t goes to infinity [24].

B. Batch Q-learning

We cannot reformulate Problems 1 and 2 into the form (23) (see Section IV). Thus, we propose an alternate $Q-$learning formulation, called *batch Q-learning*, to solve these problems. Instead of updating the Q-function after each action is taken, we wait until an entire episode $s^{[0:T]}$ is completed before updating the Q-function. The batch Q-learning procedure is summarized in Algorithm 1.

Algorithm 1 The Batch Q learning algorithm.

function BatchQLearn(Sys,probType,N_{ep},ϕ)
$Q \leftarrow$ RandomInitialization
$\mu \leftarrow$ InitializePolicy(Q)
for $n = 1$ to N_{ep} **do**
$\quad s^{[0,T]} \leftarrow$ Simulate(Sys, μ)
$\quad Q \leftarrow$ UpdateQFunction(Q,μ,$s^{0:T}$,ϕ,probType)
$\quad \mu \leftarrow$ UpdatePolicy(μ, Q)
return Q, μ

Algorithm 2 Function used to update Q function used in Algorithm 1.

function UpdateQFunction(Q,μ,$s^{0:T}$,ϕ,γ,probType)
for $n = T - \tau - 1$ to τ **do**
$\quad$ **if** probType is MaximumProbability **then**
$\quad\quad Q_{tmp}(\sigma_\tau^n, \mu(\sigma_\tau^n, T-n)) \leftarrow$
$\quad\quad \max(I(s^{n-\tau+1:n} \models \phi),$
$\quad\quad \gamma Q_{tmp}(\sigma_\tau^{n+1}, \mu(\sigma_\tau^{n+1}, T-n-1))$
$\quad$ **else**
$\quad\quad Q_{tmp}(\sigma_\tau^n, \mu(\sigma_\tau^n, T-n)) \leftarrow$
$\quad\quad \max(r(s^{n-\tau+1:n}, \phi),$
$\quad\quad \gamma Q_{tmp}(\sigma_\tau^{n+1}, \mu(\sigma_\tau^{n+1}, T-n-1))$
$\quad Q_{new}(\sigma_\tau^n, \mu(\sigma_\tau^n, T-n) \leftarrow$
$\quad (1-\alpha)Q_{tmp}(\sigma_\tau^n, \mu(\sigma_\tau^n, T-n)$
$\quad + \alpha Q(\sigma_\tau^n, \mu(\sigma_\tau^n, T-n)$
return Q_{new}

The Q function is initialized to random values and μ is computed from the initial Q values. Then, for N_{ep} episodes, the system is simulated using μ. Randomization is used to encourage exploration of the policy space. The observed trajectory is then used to update the Q function according to Algorithm 2. The new value of the Q function is used to update the policy μ. For compactness, Algorithm 2 as written only covers the case $\phi = F_{[0,T)}\psi$. The case in which $\phi = G_{[0,T)}\psi$ can be addressed similarly.

C. Convergence of Batch Q-learning

Given a formula of the form $\phi = F_{[0,T)}\psi$ and an objective of maximizing the expected robustness (Problem 2), we will show that applying Algorithm 1 converges to the optimal solution. The other three cases discussed in Section IV can be proven similarly. The following analysis is based on [19]. The optimal Q function derived from (8) is

$$Q^*(\sigma_\tau^k, a, T-k) = \quad \sum_{\sigma_\tau^{t+1}} \mathscr{P}(\sigma_\tau^t, a, \sigma^{t+1}) \max(r(\sigma_\tau^t, \psi), \\ \max_{b \in Act} \gamma Q^*(\sigma_\tau^{t+1}, b, T-t-1)).$$

(26)

This gives the following convergence result.
Proposition 2: The Q-learning rule given by

$$Q_{k+1}(\sigma_\tau^t, a^t, T-t) = \quad (1-\alpha_k)Q_k(\sigma_\tau^t, a^t, T-t) \\ + \alpha_k \max(r(\sigma_\tau^t, \psi), \\ \max_{b \in Act} \gamma Q_k(\sigma_\tau^{t+1}, b, T-t-1)),$$

(27)

converges to the optimal Q function (26) if the sequence $\{\alpha_k\}_{k=0}^\infty$ is such that $\sum_{k=0}^\infty \alpha_k = \infty$ and $\sum_{k=0}^\infty (\alpha_k)^2 < \infty$.
Proof: (Sketch) The proof of Proposition 2 relies primarily on Proposition 3. Once this is established, the rest of the proof varies only slightly from the presentation in [19]. ∎

Note that in this case, k ranges over the number of episodes and t ranges over the time coordinate of the signal.
Proposition 3: The optimal Q-function given by (26) is a fixed point of the contraction mapping H where

$$(Hq)(\sigma_\tau^t, a, T-t) = \quad \sum_{\sigma_\tau^{t+1}} \mathscr{P}(\sigma_\tau^t, a, \sigma^{t+1}) \max(r(\sigma_\tau^t, \psi), \\ \gamma \max_{b \in Act} q(\sigma_\tau^{t+1}, b, T-t-1)).$$

(28)

Proof: By (26), if H is a contraction mapping, then Q^* is a fixed point of H. Consider

$$||Hq_1 - Hq_2||_\infty = \max_{\sigma_\tau, a} \sum_{\sigma_\tau'} \mathscr{P}(\sigma_\tau, a, \sigma_\tau')(\max(r(\sigma_\tau, \psi), \\ \gamma \max_{b \in Act} q_1(\sigma_\tau', b, T-t-1)) \\ - \max(r(\sigma_\tau, \psi), \gamma \max_{b \in Act} q_2(\sigma_\tau', b, T-t-1)).$$

(29)

Define

$$q_j^*(t) = \max_{b \in Act} \gamma q_1(\sigma_\tau', b, t).$$

(30)

WOLOG let $q_1^*(T-t-1) > q_2^*(T-t-1)$. Define

$$R(\sigma_\tau') = \quad (\max(r(\sigma_\tau, \psi), q_1^*(T-t-1) \\ - \max(r(\sigma_\tau, \psi), q_2^*(T-t-1))$$

(31)

There exist 3 possibilities for the value of $R(\sigma_\tau)$.

$$r(\sigma_\tau, \psi) > q_1^*(T-t-1) > q_2^*(T-t-1) \\ \Rightarrow R(\sigma_\tau') = 0$$

(32a)

$$q_1^*(T-t-1) > r(\sigma_\tau, \psi) > q_2^*(T-t-1) \\ \Rightarrow R(\sigma_\tau') = ||q_1^*(T-t-1) - r||_\infty < \gamma ||q_1 - q_2||_\infty$$

(32b)

$$q_1^*(T-t-1) > q_2^*(T-t-1) > r(\sigma_\tau, \psi) \\ \Rightarrow R(\sigma_\tau') < \gamma ||q_1 - q_2||_\infty.$$
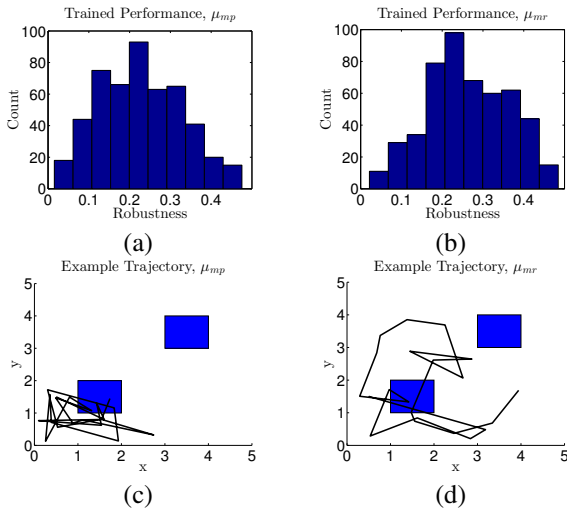
(32c)

Fig. 3: Comparison of Policies for Case Study 1. Histogram of robustness values for trained policies for solution to (a) Problem 1 and (b) Problem 2. Trajectory generated from policies for solution to (c) Problem 1 and (d) Problem 2.

Fig. 4: Comparison of Policies for Case Study 2.. The subplots have the same meaning as in Figure 3.

Thus, this means that $R(\sigma'_\tau) \leq \gamma \|q_1 - q_2\|_\infty \ \forall \sigma'_\tau$. Hence,

$$
\begin{aligned}
\|Hq_1 - Hq_2\|_\infty &= \max_{\sigma_\tau, a} \sum_{\sigma'_\tau} \mathscr{P}(\sigma_\tau, a, \sigma'_\tau) R(\sigma'_\tau) \\
&\leq \max_{\sigma_\tau, a} \sum_{\sigma'_\tau} \mathscr{P}(\sigma_\tau, a, \sigma'_\tau) \gamma \|q_1 - q_2\|_\infty \\
&\leq \gamma \|q_1 - q_2\|_\infty.
\end{aligned}
\tag{33}
$$

Therefore, H is a contraction mapping. ∎

VII. CASE STUDY

We implemented the batch-Q learning algorithm (Algorithm 1) and applied it to two case studies that adapt the robot navigation model from Example 1. For each case study, we solved Problems 1 and 2 and compared the performance of the resulting policies. All simulations were implemented in Matlab and performed on a PC with a 2.6 GHz processor and 7.8 GB RAM.

A. Case Study 1: Reachability

First, we consider a simple reachability problem. The given STL specification is

$$
\phi_{cs1} = F_{[0,20)}(F_{[0,1)}\varphi_{blue} \wedge G_{[1,4)}\neg\varphi_{blue}),
\tag{34}
$$

where φ_{blue} is the STL subformula corresponding to being in a blue region. In plain English, (34) can be stated as "Within 20 time units, reach a blue region and then don't revisit a blue region for 4 time units." The results from applying Algorithm 1 are summarized in Figure 3. We used the parameters $\gamma = 1, \alpha_t = 0.95, N_{ep} = 300$ and $\varepsilon^t = 0.995^t$, where ε^t is the probability at iteration t of selecting an action at random [1]. Constructing the τ-MDP took 17.2s. Algorithm 1 took 161s to solve Problem 1 and 184s to solve Problem 2.

The two approaches perform very similarly. In the first row, we show a histogram of the robustness of 500 trials

[1]Although the conditions $\gamma < 1$ and $\sum_{k=0}^\infty \alpha_k^2 < \infty$ are technically required to prove convergence, in practice these conditions can be relaxed without having adverse effects on learning performance
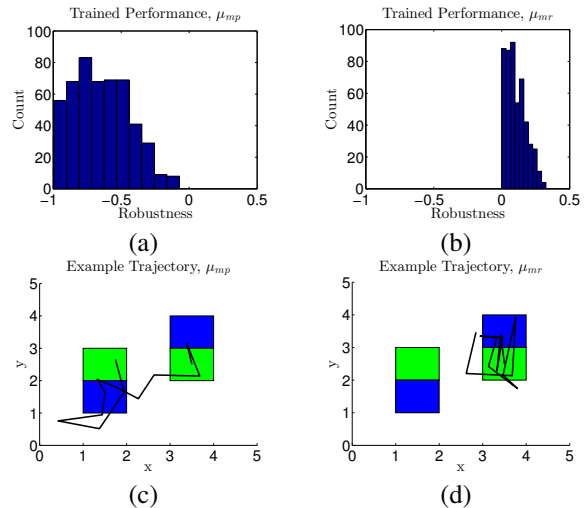
generated from the system simulated using each of the trained policies after learning has completed, i.e. without the randomization that is used during the learning phase. Note that both trained policies satisfied the specification with probability 1. The performance of the two algorithms are very similar, as the mean robustness is 0.2287 with standard deviation 0.1020 for probability maximization and 0.2617 and 0.1004,resp., for robustness maximization. In the second row, we see trajectories simulated by each of the trained policies.

The similarity of the solutions in this case study is not surprising. If the state of the system is deep within A or B, then the probability that it will remain inside that region in the next 3 time steps (satisfy ϕ) is higher than if it is at the edge of the region. Trajectories that remain deeper in the interior of region A or B also have a high robustness value. Thus, for this particular problem, there is an inherent coupling between the policies that satisfy the formula with high probability and those that satisfy the formula as robustly as possible on average.

B. Case Study 2: Repeated Satisfaction

In this second case study, we look at a problem involving repeatedly satisfying a condition finitely many times. The specification of interest is

$$
\phi_{cs2} = G_{[0,12)}(F_{[0,4)}(\varphi_{blue}) \wedge F_{[0,4)}(\varphi_{green})),
\tag{35}
$$

In plain English, (35) is "Ensure that every 4 time units over a 12 unit interval, a green region and a blue region is entered." Results from this case study are shown in Figure 4. We used the same parameters as listed in Section VII-A, except $N_{ep} = 1200, \alpha = 0.4$, and $\varepsilon^t = 0.9^t$. Constructing the τ-MDP took 16.5s. Applying Algorithm 1 took 257.7s for Problem 1 and 258.3s for Problem 2.

In the first row, we see that the solution to Problem 1 satisfies the formula with probability 0 while the solution to Problem 2 satisfies the formula with probability 1. At

first, this seems counterintuitive, as Proposition 2 indicates that a policy that maximizes probability would achieve a probability of satisfaction at least as high as the policy that maximizes the expected robustness. However, this is only guaranteed with an infinite number of learning trials. The performance in terms of robustness is obviously better for the robustness maximization (mean 0.1052, standard deviation 0.0742) than for the probability maximization (mean -0.6432, standard deviation 0.2081). In the second row, we see that the maximum robustness policy enforces convergence to a cycle between two regions, while the maximum probability policy deviates from this cycle.

The discrepancy between the two solutions can be explained by what happens when trajectories that almost satisfy (35) occur. If a trajectory that almost oscillates between a blue and green region every four seconds is encountered when solving Problem 1, it collects 0 reward. On the other hand, when solving Problem 2, the policy that produces the almost oscillatory trajectory will be reinforced much more strongly, as the resulting robustness is less negative. However, since the robustness degree gives "partial credit" for trajectories that are close to satisfying the policy, the reinforcement learning algorithm performs a directed search to find policies that satisfy the formula. Since probability maximization gives no partial credit, the reinforcement learning algorithm is essentially performing a random search until it encounters a trajectory that satisfies the given formula. Therefore, if the family of policies that satisfy the formula with positive probability is small, it will on average take the Q-learning algorithm solving Problem 1 a longer time to converge to a solution that enforces formula satisfaction.

VIII. Conclusions and Future Work

In this paper, we presented a new reinforcement learning paradigm to enforce temporal logic specifications when the dynamics of the system are *a priori* unknown. In contrast to existing works on this topic, we use a logic (signal temporal logic) whose formulation is directly related to a system's statespace. We present a novel, convergent Q-learning algorithm that uses the robustness degree, a continuous measure of how well a trajectory satisfies a formula, to enforce the given specification. In certain cases, robustness maximization subsumes the established paradigm of probability maximization and, in certain cases, robustness maximization performs better in terms of both probability and robustness under partial training. Future research includes formally connecting our approach to abstractions of linear stochastic systems.

References

[1] A. Abate, A. D'Innocenzo, and M. Di Benedetto. Approximate abstractions of stochastic hybrid systems. *Automatic Control, IEEE Transactions on*, 56(11):2688–2694, Nov 2011.

[2] C. Baier and J.-P. Katoen. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[3] T. Brazdil, K. Chatterjee, M. Chmelik, M.k, V. Forejt, J. Kretinsky, M. Kwiatkowska, D. Parker, and M. Ujma. Verification of markov decision processes using learning algorithms. In F. Cassez and J.-F. Raskin, editors, *Automated Technology for Verification and Analysis*, volume 8837 of *Lecture Notes in Computer Science*, pages 98–114. Springer International Publishing, 2014.

[4] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5):1244–1257, 2014.

[5] X. C. Ding, J. Wang, M. Lahijanian, I. Paschalidis, and C. Belta. Temporal logic motion control using actor-critic methods. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4687–4692, May 2012.

[6] A. Dokhanchi, B. Hoxha, and G. Fainekos. On-line monitoring for temporal logic robustness. In *Runtime Verification*, pages 231–246. Springer, 2014.

[7] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. *Formal Modeling and Analysis of Timed Systems*, pages 92–106, 2010.

[8] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[9] J. Fu and U. Topcu. Probably approximately correct MDP learning and control with temporal logic constraints. *CoRR*, abs/1404.7073, 2014.

[10] X. Jin, A. Donze, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. In *Proceedings of the 16th international conference on Hybrid systems: computation and contro*, pages 43–52, 2013.

[11] A. Jones, Z. Kong, and C. Belta. Anomaly detection in cyber-physical systems: A formal methods approach. In *IEEE Conference on Decision and Control (CDC)*, pages 848–853, 2014.

[12] A. Julius and G. Pappas. Approximations of stochastic hybrid systems. *Automatic Control, IEEE Transactions on*, 54(6):1193–1203, June 2009.

[13] M. Kamgarpour, J. Ding, S. Summers, A. Abate, J. Lygeros, and C. Tomlin. Discrete time stochastic hybrid dynamic games: Verification and controller synthesis. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6122–6127, 2011.

[14] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 273–282. ACM, 2014.

[15] M. Lahijanian, S. Andersson, and C. Belta. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *Robotics, IEEE Transactions on*, 28(2):396–409, April 2012.

[16] M. Lahijanian, S. B. Andersson, and C. Belta. Approximate markovian abstractions for linear stochastic systems. In *Proc. of the IEEE Conference on Decision and Control*, pages 5966–5971, Maui, HI, USA, Dec. 2012.

[17] M. Lahijanian, S. B. Andersson, and C. Belta. Formal verification and synthesis for discrete-time stochastic systems. *IEEE Transactions on Automatic Control*, 6(8):2031–2045, 2015.

[18] R. Luna, M. Lahijanian, M. Moll, and L. E. Kavraki. Asymptotically optimal stochastic motion planning with temporal goals. In *Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, 03/08/2014 2014.

[19] F. S. Melo. Convergence of q-learning: a simple proof. http://users.isr.ist.utl.pt/ mtjspaan/readingGroup/ProofQlearning.pdf.

[20] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. Model predictive control with signal temporal logic specifications. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, pages 81–87, 2014.

[21] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia. A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications. *CoRR*, abs/1409.5486, 2014.

[22] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[23] M. Svorenova, J. Kretínský, M. Chmelik, K. Chatterjee, I. Cerná, and C. Belta. Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games. In *Hybrid Systems Computation and Control (HSCC) 2015*, volume abs/1410.5387, 2015. (To appear).

[24] J. N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16(3):185–202, 1994.